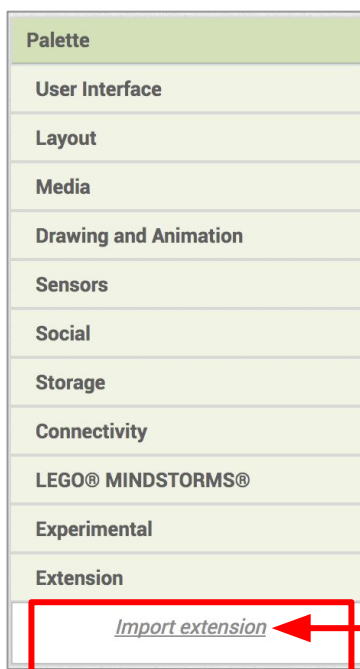
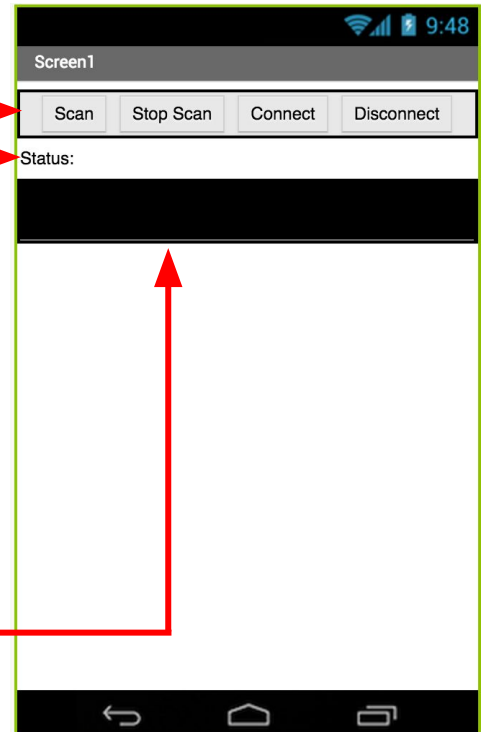


App Inventor + IoT: Basic Bluetooth Connection Setup

Start a new project in [App Inventor](#) and name it BasicIoTSetup.

First, we need to set up some buttons to find and connect to our Arduino over Bluetooth.

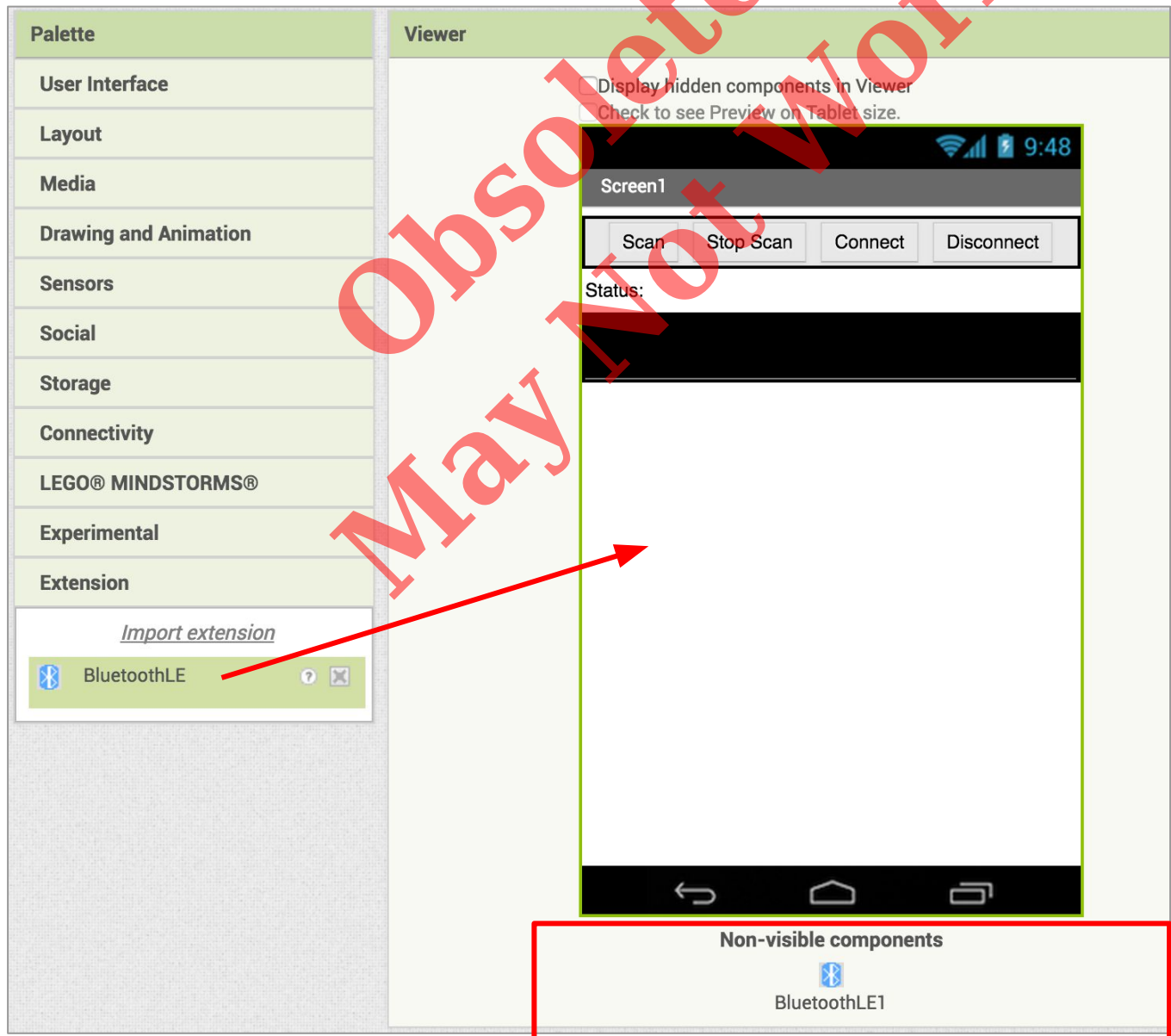
- Drag a *HorizontalArrangement* from the Layout drawer in the Palette and add 4 Buttons to it.
- Rename the buttons: **ButtonScan**, **ButtonStopScan**, **ButtonConnect**, and **ButtonDisconnect**.
- Change their text to "Scan", "Stop Scan", "Connect", and Disconnect".
- Below the *Horizontal Arrangement* add a Label. Rename it **LabelStatus** and change its text to "Status: ".
- Drag in a *ListView* below **LabelStatus** and rename it **ListBLE**.



Next, we need to install the BluetoothLE extension.

- Download the edu.mit.appinventor.ble.aix extension to your computer.
- In the Palette, click on Extension at the bottom and then on "Import extension" and then "Choose File".
- Find the extension on your computer and upload it.

Add the BluetoothLE extension by dragging it onto the Viewer.



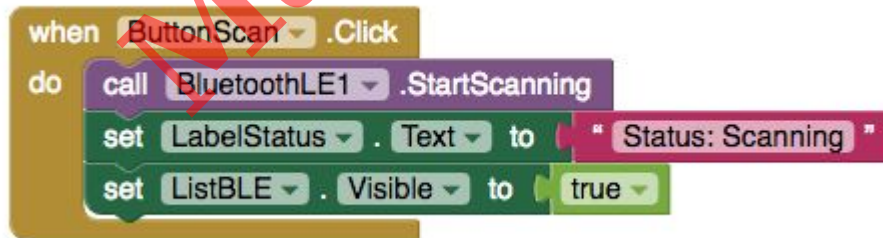
After it is are dragged onto the Viewer it will appear below the main screen, since it is a non-visible component.

Switch to the Blocks Editor view

We want to set up the app to scan for available Bluetooth devices. To do this, we will use the ButtonScan button to set the Bluetooth component to start scanning, and change the status label.

- From the Blocks pane, click on ButtonScan, and drag out **when ButtonScan.Click**.
 - from the BluetoothLE1 drawer, add **call BluetoothLE1.StartScanning**.
 - from the LabelStatus drawer, add **set LabelStatus.Text to**
 - from the Text drawer, add a text block and type in **"Status: Scanning"**.
 - from the ListBLE drawer, add **set ListBLE.Visible**.
 - From the Logic drawer, drag out a **true** block and snap to **set ListBLE.Visible**.

Note: We do this so we can hide the list later, because it can get very long if there are a lot of Bluetooth devices nearby.



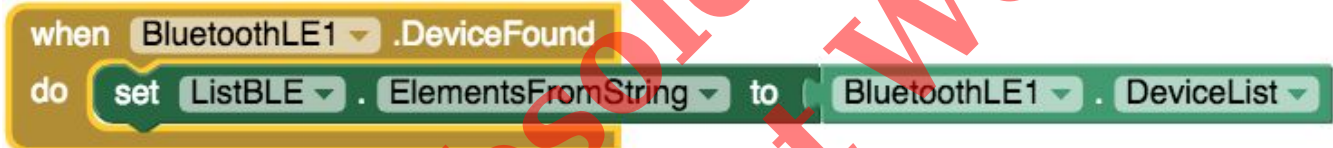
Next, we'll have the app stop scanning and change the status label when we press ButtonStopScan.

- From the Blocks pane click on ButtonStopScan, and drag out **when ButtonStopScan.Click**.
 - from the BluetoothLE1 drawer, add **call BluetoothLE1.StopScanning**.
 - from the LabelStatus drawer, add **set LabelStatus.Text to**.
 - from the Text drawer, add a text block and type in **"Status: Stopped Scanning"**



We need to populate the device list with all the available Bluetooth devices.

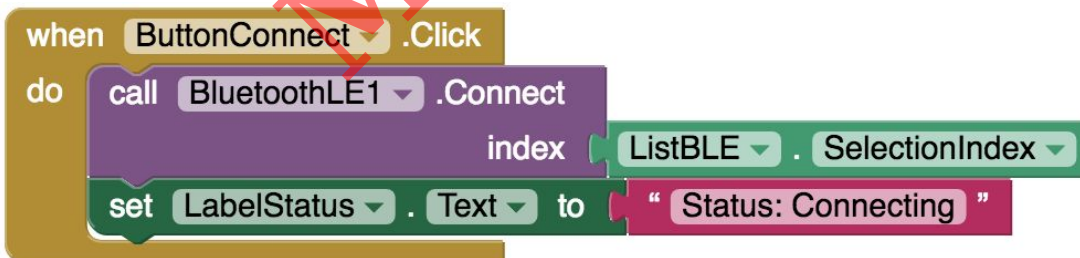
- From the Blocks pane, click on BluetoothLE1, and drag out **when BluetoothLE1.DeviceFound**.
 - from the ListBLE drawer, add **set ListBLE.ElementsFromString** to
 - From the BluetoothLE1 drawer, add **BluetoothLE1.Devicelist**.



```
when BluetoothLE1 .DeviceFound
do set ListBLE . ElementsFromString to BluetoothLE1 . DeviceList
```

Now, we'll use the app to connect to the Arduino over Bluetooth.

- From the Blocks pane, click on ButtonConnect, and drag out **when ButtonConnect.Click**.
 - from the BluetoothLE1 drawer, add **call BluetoothLE1.Connect** index.
 - from the ListBLE drawer, add **ListBLE.SelectionIndex**.
 - from the LabelStatus drawer, add **set LabelStatus.Text** to
 - from the Text drawer, add a text block and type **"Status: Connecting"**.



```
when ButtonConnect .Click
do call BluetoothLE1 .Connect index ListBLE . SelectionIndex
set LabelStatus . Text to "Status: Connecting"
```

Let's set it up so we know when the app has successfully connected to the Arduino:

- From the Blocks pane, click on BluetoothLE1, and drag out **when BluetoothLE1.Connected**.
 - from the LabelStatus drawer, add **set LabelStatus.Text** to
 - From the Text drawer, add a text block and type **"Status: Connected"**
 - from the ListBLE drawer, add **set ListBLE.Visible** to
 - From the Logic drawer, add a **false** block.

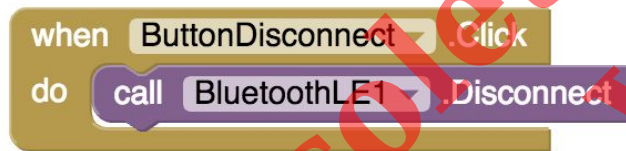
Note: We do this to hide the list. As we mentioned above, it can get very long if there are a lot of Bluetooth devices nearby.



```
when BluetoothLE1 .Connected
do set LabelStatus . Text to "Status: Connected"
set ListBLE . Visible to false
```

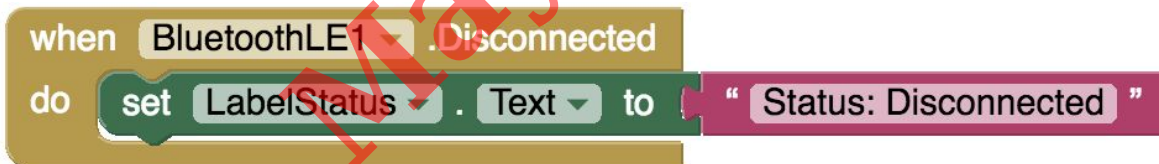
Next we want to be able to disconnect from the Bluetooth device.

- From the Blocks pane, click on ButtonDisconnect, and drag out **when ButtonDisconnect.Click**
 - from the BluetoothLE1 drawer, add **call BluetoothLE1.Disconnect**.



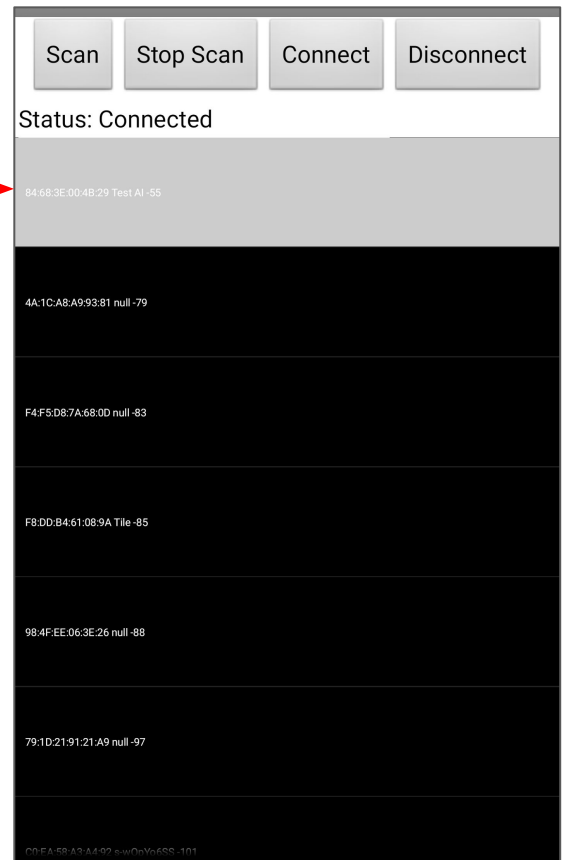
We also want to know when the Bluetooth device successfully disconnects (to know pressing the button above worked)

- From the Blocks pane, click on BluetoothLE1, and drag out **when BluetoothLE1.Disconnected**.
 - from the LabelStatus drawer, add **set LabelStatus.Text to**
 - from the Text drawer, add a text block and type **"Status: Disconnected"**.



Now let's test out our app using the MIT AI2 Companion. Once you've connected your device to your computer, test the app using the following steps:

- Click the **Scan** button.
You should see a list of BLE devices.
- When you see your device, click **Stop Scan**.
- Click on your device name in the list.
- Click **Connect**.
If your device successfully connects your **LabelStatus** should change to "Status: Connected".



Note: This tutorial only connects your app to your IoT Bluetooth device. To do something fun with the device, you will want to set it up and then try adding sensors (light, moisture, etc) to control and/or display information.